MICRO FOCUS

Silk Performer 18.0

Citrix Tutorial

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2017-06-07

目次

Citrix XenApp のサポート

Silk Performer では、Citrix XenApp セッションおよびアプリケーションの仮想化テクノロジを介してホストされるアプリケーションをテストするための、記録と再生のサポートを提供しています。

Citrix を使用すると、ネットワークおよびインターネットで共有アプリケーションへのリアルタイム アクセスを簡単に行うことができます。 Citrix 対応アプリケーションへのリモート アクセスは、DSL、T1、ISDN、またはダイヤルアップで行うことができます。 Citrix を使用すると、複数のユーザーが共有アプリケーションを同時に実行できます。 Citrix プラグイン (クライアント) とサーバー間の通信では、ユーザー入力 (キーボード/マウス) とスクリーンショットがやりとりされます。

Silk Performer では、Citrix サーバーの記録と Citrix にホストされるアプリケーションの記録の両方がサポートされます。 Citrix プラグインにより、特定のアプリケーションおよび複数のデスクトップ ウィンドウへのアクセスが可能になります。

Citrix Web インターフェイスを使用すると、ユーザーは、ホスト コンピュータに直接接続しなくても、ホストされている XenApp アプリケーションおよび XenDesktop 仮想デスクトップにアクセスできます。 ユーザーは、Web ブラウザまたは Citrix オンライン プラグインのいずれかを介してこれらのリソースにアクセスします。

Silk Performer では、以下の 2 つの Citrix プロジェクト タイプを提供しています。

- *Citrix*。Citrix プラグイン (クライアント) を介してアクセスする、ホストされているアプリケーションのテストに使用
- *Citrix Web* インターフェイス。Citrix Web インターフェイスを介して公開されている Citrix 対応アプリケーションのテストに使用。

PDF 形式の Citrix XenApp チュートリアルでは、Citrix 対応アプリケーションのテストの全過程をたどります。 このチュートリアルは、**スタート** > **すべてのプログラム** > **Silk** > **Silk Performer18.0** > **ドキュメント** > **チュートリアル** > **Citrix XenApp** の Silk Performer から利用できます。

注: Silk Performer のヘルプには、Citrix XenApp チュートリアルに含まれているすべてのコンテンツが含まれています。

Citrix Program Neighborhood

以前のバージョンの Citrix は、アプリケーションの配布で Web インターフェイスに依存する代わりに Windows ベースの Citrix Program Neighborhood を使用します。 Citrix Program Neighborhood を使用すると、ユーザーは Citrix XenApp オンデマンドのアプリケーション配信サーバーによる、利用可能なすべてのアプリケーションのリストを表示できます。 Citrix ブラウザ サービスを経由して Citrix Program Neighborhood を起動するたびに、利用可能なリソースが自動的に列挙されます。

重要: Citrix Program Neighborhood 経由で配布されるアプリケーションの場合、システム上でこれを起動するには Citrix 公開アプリケーションの記録サービスを設定する必要があります。これを行わないと、Citrix Program Neighborhood でアプリケーション リストが更新されません。

プロジェクトを定義する

Citrix の負荷テストを実施する際の最初のステップは、Silk Performer プロジェクトの基本設定を定義することです。 その後で、前提となる XenApp サーバーの構成を実施する必要があります。

Citrix クライアント ソフトウェアの構成

Citrix プロジェクトを定義する前に、Citrix クライアント ソフトウェア (Citrix Receiver) をインストールし、Citrix サーバーがタイムアウトを処理する方法を構成する必要があります。Citrix クライアント ソフトウェアをダウンロードするには、*http://www.citrix.com* に移動します。

Citrix サーバーの構成は、ご使用のバージョンによります。以下は Citrix Server 4.0 の構成手順です。

注: Citrix サーバー バージョン 4.5 以降を構成するには、**スタート** > **プログラム** > **管理ツール** > **ターミナル サービス構成** に移動し、**ica-tcp** オプションをダブルクリックします。

1. **スタート** > **プログラム** > **Citrix** > **管理ツール** に移動し、**Citrix コネクション構成ツール** を開始します。
2. **ica-tcp** 接続をダブルクリックします。
3. 以下のダイアログ ボックスで、**詳細設定** をクリックします。**コネクションの詳細設定** ダイアログ ボックスが表示されます。
4. **切断タイムアウト** 制限を設定します。推奨のタイムアウトは 1 分です。
5. **アイドル タイムアウト** 制限を設定します。推奨のタイムアウトは 5 分です。
6. **接続が切断またはタイムアウトしたときの処理** ドロップ リストで **リセット** を選択します。

   このオプションを選択しないと、接続の切断またはタイムアウトにより、セッションは再生停止後も開いたままとなります。これにより、前のセッションで停止した箇所からスクリプトが継続するため、ユーザーがセッションに次にログインするときに再生上の問題が生じます。そのような場合は、セッションを手動で終了する必要があります。

Citrix XenApp プロジェクトを定義する

1. Silk Performer ワークフロー バーの **ここから開始する** をクリックします。

   注: 別のプロジェクトが既に開いている場合に、メニュー バーから **ファイル** > **プロジェクトの新規作成** を選択すると、現在開いているプロジェクトを閉じるように確認メッセージが表示されます。

   ワークフロー - プロジェクトの概要設定 ダイアログ ボックスが開きます。
2. **名前** テキスト ボックスに、プロジェクトの名前を入力します。
3. オプションのプロジェクト説明を **説明** に入力します。
4. **種類** メニュー ツリーで、**ターミナル サービス** > **Citrix** または **ターミナル サービス** > **Citrix Web インターフェイス** を選択します。

   Citrix のアプリケーションの種類は Silk Performer Citrix Recorder を使用して、Citrix オンラインまたはオフライン プラグインからアクセスする Citrix 対応アプリケーションの配信をテストします。

   Citrix Web インターフェイス のアプリケーションの種類は、Citrix Web インターフェイスからアクセスする Citrix 対応アプリケーションの配信のテストに使用されます。
5. **次へ** をクリックし、設定に基づいてプロジェクトを作成します。

ワークフロー - スクリプトの作成 ダイアログ ボックスが表示されます。

Citrix プラグイン テスト スクリプトを作成する

直接 Citrix XenApp サーバーに接続してテスト スクリプトを作成する Citrix XenApp プラグイン (クライアント) を介したユーザー アクションを記録するための最も簡単な方法は、Silk Performer Recorder を使用することです。この Recorder は、トラフィックの取得と記録、およびテスト スクリプトの生成に使用される Silk Performer のエンジンです。

Silk Performer Recorder は、テスト対象の Citrix XenApp プラグインとサーバーの間のトラフィックを取得して記録します。 記録が終了すると、Silk Performer Recorder は、記録されたトラフィックに基づい

てテスト スクリプトを自動生成します。 スクリプトは、Silk Performer のスクリプト言語である Benchmark Description Language (BDL) で記述されます。

1. ワークフロー バーの **スクリプトの作成** をクリックします。 **ワークフロー - スクリプトの作成** ダイアログ ボックスが表示されます。
2. **アプリケーション プロファイル** リストで Silk Performer Citrix Recorder を選択し、Silk Performer Citrix Recorder アプリケーションを介して Citrix XenApp 対応アプリケーションを記録します。

   このアプリケーション プロファイルは、直接 Citrix XenApp サーバーに接続する Citrix プラグインのテストに適しています。
3. 省略可能：サーバー接続パラメータを定義する ICA ファイルがある場合は、**コマンド ライン** フィールドに ICA ファイルのフルパスを入力します。
4. **記録の開始** をクリックします。
5. 次に Silk Performer Citrix Recorder と共に、Silk Performer Recorder が最小化されて開きます。

   起動時に正しいログイン資格情報が使用不可の場合は、**接続** ダイアログが開きます。 **接続** ダイアログには、Silk Performer Citrix Recorder の左上隅にある **接続** ボタンからもアクセスできます。

   注: サーバー接続パラメータを定義する ICA ファイルがある場合は、**ICA ファイル** を選択し、ICA ファイルを参照して **ログイン** フィールドをすべてスキップします。
6. **サーバー** フィールドの **接続** ダイアログで、記録する Citrix XenApp サーバーの名前を入力します。
7. テスト対象のサーバーの **ユーザー名** フィールド、**パスワード** フィールド、および **ドメイン** フィールドに入力します。
8. **アプリケーション** フィールドに、ホストされるアプリケーションの名前を入力します。
9. **クライアント名** フィールドを使用して、セッションのクライアント名を指定することができます。 Citrix Recorder のデフォルトのクライアント名は、SP_Recorder です (Citrix Player のデフォルトのクライアント名は、SP_User_x です)。 記録用に異なるクライアント名を指定すると、CitrixSetClientName 関数がスクリプトに挿入されます。 この場合、クライアント名の値をカスタマイズする必要があります。カスタマイズしないと、すべてのユーザーが同じクライアント名を使用することとなり、再生の問題が発生する場合があります。
10. 記録に必要な **色の解像度** を選択します。
11. 記録に必要な画面の **解像度** を選択します。

    注: スクリプトの記録後は、この解像度を変更しないでください。異なる解像度設定で再生すると、位置変更された要素上でマウスが動作しなくなることがあるためです。
12. **接続** をクリックして Citrix XenApp セッションを開始します。
13. テスト中に仮想ユーザーが行う操作と同じ方法で、Citrix Recorder で共有デスクトップとやり取りします。 たとえば、リンクをクリックしたり、アプリケーションを開いたり、データを入力したりすることができます。 アクションは、Citrix Recorder によってキャプチャされ、記録されます。

    Citrix Recorder ではセッション共有がサポートされ、既存のセッションで追加の公開アプリケーションを (Citrix Recorder ツールバーで **アプリケーションの実行** をクリックして) 起動できます。 **ウィンドウの選択** をクリックしてアプリケーションを切り替えます。

    次の Citrix XenApp セッションの例には、Excel の起動、新しいデータ値の入力、AutoSum 式の挿入、画面領域の選択、AutoSum 式の編集、および Excel の終了にマウスとキーボードを使用する、単純な Excel の計算が含まれています。
14. Citrix XenApp セッションの記録が終了したら、Silk Performer Citrix Recorder の **停止** をクリックします。 **スクリプトの生成** 進捗状況ウィンドウが開き、その後で **名前を付けて保存** ダイアログが開きます。
15. スクリプトを意味のある名前で保存します。
16. 記録された操作に基づいて新しく生成されたテスト スクリプトが、Silk Performer のスクリプト エディタ ウィンドウに表示されます。

Citrix Web インターフェイス セッション (NFuse)

Citrix Web インターフェイス ソフトウェア (以前の NFuse) では、Citrix アプリケーション サーバー ソフトウェアがホストしている Java、UNIX、Windows のアプリケーションへのアクセスを提供します。 Citrix では、ホストされているアプリケーションのサーバー側のコントロールが提供されている一方、アプリケーションが Citrix Web インターフェイスにより、Web ブラウザ インターフェイス (Internet Explorer、バージョン 5.5. 以降) から アクセス可能になります。

Citrix Web インターフェイスに関する技術サポートおよび質問は、*http://support.citrix.com* を参照してください。

Citrix Web インターフェイス プロジェクトを定義する

1. Silk Performer ワークフロー バーの **ここから開始する** をクリックします。

   注: 別のプロジェクトが既に開いている場合に、メニュー バーから **ファイル** > **プロジェクトの新規作成** を選択すると、現在開いているプロジェクトを閉じるように確認メッセージが表示されます。

   ワークフロー - プロジェクトの概要設定 ダイアログ ボックスが開きます。
2. **名前** テキスト ボックスに、プロジェクトの名前を入力します。
3. オプションのプロジェクト説明を **説明** に入力します。
4. **種類** メニュー ツリーで、**ターミナル サービス** > **Citrix Web インターフェイス** を選択します。
5. **次へ** をクリックし、設定に基づいてプロジェクトを作成します。

ワークフロー - スクリプトの作成 ダイアログ ボックスが表示されます。

Citrix Web インターフェイス テスト スクリプトを作成する

Citrix Web インターフェイス セッションのテスト スクリプトを作成する最も簡単な方法は、Silk Performer Recorder を使用することです。この Recorder は、トラフィックを取得して記録し、テスト スクリプトを生成する Silk Performer のエンジンです。

Silk Performer Recorder は、テスト対象の Citrix Web インターフェイス クライアント アプリケーション (Internet Explorer、バージョン 5.5 以上) とサーバーの間のトラフィックを取得して記録します。 記録が終了すると、Silk Performer Recorder は、記録されたトラフィックに基づいてテスト スクリプトを自動生成します。 スクリプトは、Silk Performer のスクリプト言語である Benchmark Description Language (BDL) で記述されます。

1. ワークフロー バーの **スクリプトの作成** をクリックします。 **ワークフロー - スクリプトの作成** ダイアログ ボックスが表示されます。
2. **アプリケーション プロファイル** リスト から、Citrix Web インターフェイス を選択します。

   Citrix Web インターフェイス アプリケーション プロファイルは、Citrix Web インターフェイス/NFuse セッションのテストにのみ適しています。
3. **記録の開始** をクリックします。
4. 次に Internet Explorer と共に、Silk Performer Recorder が最小化されて開きます。 Internet Explorer **アドレス** フィールドに Citrix サーバーの名前を入力し、**Enter** をクリックします。 記録中に発生したアクションのレポートを表示するには、Recorder ツールバーの **GUI サイズの変更** をクリックして Silk Performer Recorder ダイアログを最大化します。
5. Citrix Web インターフェイスにログインするには、Citrix Web インターフェイス ログイン画面で **ユーザー名**、**パスワード**、および **ドメイン** を入力します。 これらの情報がない場合は、システム管理者に問い合わせてください。
6. **ログイン** をクリックします。
7. アプリケーション ポータルが表示されます。 このポータルには、共有用に公開されているアプリケーションがあります。 記録対象のホストされているアプリケーションを選択します。

8. ホストされているアプリケーションが Silk Performer Citrix Recorder に表示されます。 テスト中に仮想ユーザーが行う操作と同じ方法で、Citrix Recorder で共有アプリケーションとやり取りします。 操作は、Citrix Recorder によってキャプチャされ、BDL スクリプト内に生成されます。
9. アプリケーションを終了すると、Citrix セッションの接続が解除され、記録されたスクリプトを保存できます。

記録された Citrix Web インターフェイス セッションの BDL スクリプトは、少数の Silk Performer Web 関数を含むマルチプロトコル スクリプトです。

Citrix スクリプト関数

- 画面の同期に使用される CitrixWaitForWindowCreation は、最も重要な Citrix 関数です。

  同期で考慮される最初のパラメータは、ウィンドウ キャプションです。 再生中にウィンドウ キャプションが、再生中に記録されたウィンドウのキャプションと一致すると返された場合、検証は成功です。

  注: ワイルドカード (*) は、ウィンドウ キャプションの最初または最後に指定できます。

  2 番目に考慮されるパラメータは、一致パラメータです。 通常これは、ウィンドウ キャプション文字列の正確で大文字と小文字まで区別する比較を行います。

  3 番目のパラメータはウィンドウのスタイルです。 ウィンドウ キャプションの名前が使用できない場合、ウィンドウの特定にはスタイル名の一部が使用されます。 スタイルは、記録中にウィンドウが最大化されたか、されていないかを反映します。

  考慮する 4 番目のパラメータは、ウィンドウの位置です。 これは、最大化されたウィンドウが -4，-4 の位置にあるため、負の数となります。 このパラメータは、**ウィンドウの位置を強制する** プロファイル設定によって制御できます。 このプロファイル設定が有効にされると、ウィンドウは自動的に記録中と同一の位置に移動します。

  5 番目のパラメータはウィンドウ サイズです。 再生中、ウィンドウを記録中と同じサイズに変更する必要があります。
- CitrixWaitForLogon は、Citrix クライアントがログオンに成功、または指定したタイムアウト期間が終了するまで待機します。
- CitrixWaitForScreen は画面領域をキャプチャし、指定した条件 (通常は記録時に取得されたハッシュ値) を参照して確認します 。 条件が一致せず、タイムアウト期間が終了すると、関数呼び出しは失敗します。 CitrixWaitForScreen 関数はまた、画面表示を使用して、後続のユーザー アクションを同期します。 提供されたパラメータに基づいて、この関数は、指定した画面領域のイメージが変更されて、記録時に取得されたハッシュ値と一致または不一致となるまで、待機します。
- CitrixGetScreen は選択した領域のスクリーンショットを取得し、結果ディレクトリのファイルにスクリーンショットを書き込みます。 ファイル名は、省略された場合、ユーザー ID およびイメージのハッシュ値によって自動的に生成されます。
- CitrixGetScreenHash は、選択した画面のハッシュ値を取得します。
- CitrixSetOption により、ネットワーク プロトコル仕様、SpeedScreen 待機時間の短縮、データ圧縮、イメージ キャッシュ、マウス/キーボードのタイミングおよびイベントのキュー、 クライアントの切断、同期タイムアウト、思考時間、TrueLog キャプチャ、ウィンドウ位置の強制など、特定のオプションが設定されます。
- CitrixWaitForWindow は、指定したウィンドウ イベント (nEvent パラメータで指定) が発生するまで待機します。 そのようなイベントには、指定したウィンドウの有効化、破棄、キャプション変更があります。 選択したイベントがキャプション変更である場合、指定したウィンドウのキャプション変更が一致すると、関数が満たされます。 キャプションは、sCaption パラメータと nMatch パラメータを使用して明示的に指定できます。
- CitrixKeyString は、印刷可能な文字の入力のための標準的な関数です。
- CitrixMouseClick によりマウスが指定した位置に移動し、指定したボタンが押されます。 必要に応じて、マウスは、ボタンが押されている間に移動するように指定することもできます。 キー修飾子 (Ctrl および Alt など) を使用できます。

画面の同期と検証

Silk Performer は、Citrix XenApp サーバーをホストとするアプリケーションのビットマップとウィンドウの検証をサポートしています。 画面の同期は、再生された Citrix コンテンツを検証するための手段を提供します。 画面の同期は、標準的なスクリプトの検証とは異なり、入力値ではなく、ウィンドウと画面領域の表示の検証が行えます。 また、画面の同期は、後で行われるスクリプトのカスタマイズの時ではなく、スクリプトの記録中に挿入されます。 画面の同期は、ブラウザや類似のインターフェイスで表示される後続のユーザー入力の同期で、特に有用です (たとえば、Citrix アプリケーション ウィンドウ)。そのようなアプリケーションでは、ユーザー入力やテキスト表示などのウィンドウ イベントでの自動的な同期がサポートされていないためです。

注: レスポンス データの検証は、Citrix テストではサポートされていません。

Silk Performer は、ハッシュ値に依存して、記録されたビットマップに対して再生されたビットマップを検証します。 ハッシュ値は、サイズ、位置、解像度、色の解像度などのビットマップ仕様を反映する、コンピュータで読み込み可能な値です。

注: 記録時に取得されたハッシュ値に対して再生画面領域を検証するためには、記録時に使用した色の解像度と同じ色の解像度を、再生中も使用することが必要です。 1 ピクセルの変更でもハッシュ値の変更となり、再生コンテンツの表示が記録したコンテンツと異なる結果となる可能性があるため、仕様が維持されないと、スクリプトが失敗します。

注: 記録時にウィンドウを最大化していた場合、再生時も最大化しておく必要があります。 これは、再生がウィンドウの状態を変更できないためです (ウィンドウのサイズ変更と移動のみできます)。 そのため、ウィンドウの状態が変わっていると (たとえば、最大化されていた ものが 元に戻されていた 場合)、スクリプトのユーザー入力の一部が変わってしまう可能性があります (たとえば、**元に戻す** ボタンがクリックされた場合)。 再生では、ユーザーは画面の同じ位置をクリックしますが (このときは **最大化** ボタンになっているため)、 結果として異なる操作が実行され、その後の CitrixWaitForWindowRestore 関数が失敗します。

テキストと画面の同期

CitrixWaitForWindow() および CitrixWaitForWindowCreation() などのウィンドウ同期は、アプリケーションとの同期に非常に適しています。 アプリケーションで追加のユーザー入力の準備が整うまでスクリプトが待機するようにするには、アプリケーションと同期することが重要です。 同期を使用しないと、スクリプトがアプリケーションでオーバーランすることがあります。 また、同期をタスクが完了した時点の基準とすることによって、レスポンス時間を効果的に測定することができます。

テキストの同期

アプリケーション上で実行される多くのタスクは、ウィンドウの表示/非表示や作成/破棄をせずに、単純に画面の一部を変更するだけです。 そのような場合、Silk Performer のテキスト同期機能を使用します。

スクリプトの記録後、TrueLog Explorer テキストの同期 オプションにより、テキストの同期点を視覚的に追加できます。 テキストの同期は、組み込み OCR によって動作します。

画面の同期

Silk Performer では、2 種類の画面同期が用意されています。内容の変化を待つ および 内容の一致を待つです。 この形式の同期は、ウィンドウの参照および 2 セットの x/y 座標を必要とします。 ただし、予期しないイベントによりウィンドウの座標が移動してしまう可能性があるため、できる限りテキストの同期を使用し、画面の同期は、比較するテキストがない場合にのみ使用する必要があります。

内容の変化を待つは、画面の選択した部分が変更されるまで待機するのに対し、内容の一致を待つは、画面の選択した部分が記録中に定義された事項に一致するまで待機します。

注: 画面の同期点は、テストケースの記録中に挿入する必要があるのに対し、テキストの同期点は、TrueLog Explorer によって後で挿入できます。

記録中の画面領域の同期を生成する

画面の同期は、CitrixWaitForScreen 関数で行うことができますが、これは Recorder では自動的にスクリプト化されません。 これらの関数は、記録時に **画面領域** ダイアログ ボックスによって挿入されます。 CitrixWaitForScreen 関数では、再生ビットマップと記録ビットマップを比較して、それらが同一かどうかを判別します。 ハッシュ値は、実際のビットマップとは異なり、イメージを比較するために使用されます。 これにより、再生中のリソース消費が抑えられます。

注: 画面領域の同期は、Silk Performer Citrix Recorder によってのみ可能です。

1. Citrix セッションを記録してテスト スクリプトを作成します。
2. 記録中に Silk Performer Citrix Recorder で **領域の選択** をクリックします。
3. クリックしてカーソルをドラッグし、ビットマップ同期を生成する画面領域を選択します。

   注: 1 ピクセルの違いでも同期処理の失敗につながることがあるため、テキスト検証では、必要とする画面領域を最小限にして選択することをお勧めします。 最小限にしないと、予期しない画面の相違 (たとえば、ツールバーの無効化) が発生して、検証結果に影響を及ぼす場合があります。
4. **選択** ダイアログ ボックスが開きます。 画面領域の座標を記述する方法を指定します (絶対座標をスクリプト記述する、ウィンドウに対する相対座標をスクリプト記述する、または 座標ではなくフル ウィンドウを使用する)。 ウィンドウが最大化されている場合、絶対座標と相対座標の間に事実上違いはありません。 ウィンドウが最大化されていない場合、相対座標は Citrix Recorder ウィンドウの左上隅から測定し、絶対座標は固定された x/y 座標を使用します。
5. Citrix XenApp Player が再生中に待機する必要のある **内容の照合** タイプ を指定します (内容の一致、内容の不一致、または 内容の変化)。
6. **OK** をクリックして、同期をテスト Citrix XenApp スクリプトに追加します。

OCR による検証と解析

Silk Performer の光学文字認識 (OCR) サポートを利用すると、キャプチャされたアプリケーション状態のスクリーンショット内のテキスト値が認識されるため、セッションに依存する検証や解析が容易になります。

検証関数と解析関数が、スクリプト記録後に TrueLog Explorer により追加されます。

ウィンドウの位置と状態

ウィンドウの位置と状態 (最大化/最小化) は、正確な再生を保証するために非常に重要です。TrueLog Explorer では、選択されているテキストが、個々のウィンドウではなくデスクトップに対する相対座標から読み取られるように画面座標がスクリプト記述されるからです。従って、再生時に、記録時と異なる位置にウィンドウが表示されると、指定したテキストを OCR 操作で見つけることができません。絶対位置を指定できない場合は、ウィンドウに対する相対座標を使用して、手動でスクリプトを更新する必要があります。

OCR 検証関数を追加する

OCR サポートを利用すると、Silk Performer は認識可能なテキスト値を変数に格納できるようになり、Citrix テストでのセッション依存の検証が容易になります。

概要

OCR による文字列検証は、CitrixVerifyText API 呼び出しを使用して実現されます。これらの関数は、スクリプトのカスタマイズ時に TrueLog Explorer によって挿入されます。CitrixVerifyText 関数は、再生ビットマップ内のテキスト文字列を比較して、それらが同じかどうかを判別します。

CitrixParseText 関数は、テキストを解析するために使用できます。これらの API 呼び出しは、その他の Silk Performer 解析関数と同様に動作します。

OCR では、さまざまなフォントやテキスト スタイルを認識するためにパターン データベースが使用されます。フォント データベースは、OCR を実行する前に生成しておく必要があります。

Citrix TrueLogs では、検証と解析の API 呼び出しがツリー ビューで表示されます。

注: 頻繁に変えられる画面イメージに対する作業では、再生のタイミングが重要であるため、OCR 操作は、安定したコンテンツに対して実行する必要があります。ウィンドウ イベントでの同期を行うと、画面の更新がわずかに遅れることがあります。これによって、タイミングに依存した結果が生じます。したがって、各 OCR 検証/解析関数の前に、待機または CitrixWaitForScreen 関数の呼び出しのいずれかのスクリプトを記述することをお勧めします。

以下の 2 つの画面例は、TryScript の実行後の検証関数と解析関数の出力を示しています。

OCR 検証関数の生成

1. Silk Performer から Citrix セッションを記録します。
2. **TryScript** ダイアログ ボックスの **アニメーション** チェックボックスをオンにして TryScript を実行します。 TrueLog Explorer が開きます。
3. TryScript の実行が完了したら、テキスト検証の対象となるページのビットマップ画面キャプチャを含む CitrixSynchronization API ノード (または子ノード) を選択します。
4. 画面上でクリックしてカーソルをドラッグし、検証に使用するテキストが含まれているページ領域を選択します。
5. 選択した領域で右クリックし、コンテキスト メニューから **選択したテキストの検証** を選択します。
6. **テキスト検証関数の挿入** ダイアログ ボックスが開きます。 選択されたテキストが 定数値編集ボックスにプリロードされ、定数値ラジオ ボタンがデフォルトで選択されます。

   定数値に対する検証だけでなく、既存または新規のパラメータと照らして検証することもできます。 パラメータを検証するには、パラメータ ラジオ ボタンを選択します。 パラメータが既に存在する場合は、[**...**] をクリックすると、そのパラメータを参照して選択できます。 パラメータが存在しない場合は、[**...**] をクリックすると、新規パラメータの作成に使用できる **パラメータ ウィザード** が起動されます。
7. **選択した四角形内のテキストが以下に** ドロップ リストから 等しい または 等しくない を選択します。
8. 検証が、**大文字/小文字を区別する** かどうか、**空白を無視する** かどうかを指定します。
9. ダイアログ ボックスの **深刻度** エリアで、検証が否定的な結果を返した場合に発生させる深刻度を指定します([エラー]、[警告]、[情報]、または [カスタム])。
10. **OK** をクリックします。
11. 確認のためのダイアログ ボックスが表示されます。 **OK** をクリックして、Citrix テスト スクリプトに OCR 検証関数を追加します。

OCR 解析関数を生成する

1. Silk Performer から Citrix セッションを記録します。
2. **TryScript** ダイアログ ボックスの **アニメーション** チェックボックスをオンにして TryScript を実行します。 TrueLog Explorer が開きます。
3. TryScript の実行が完了したら、テキスト解析の対象となるページのビットマップ 画面キャプチャを含む CitrixSynchronization API ノード (または子ノード) を選択します。
4. クリックしてカーソルをドラッグし、解析するテキストを含むページ領域を選択します。
5. 選択した領域で右クリックし、**選択したテキストを変数に変換** を選択します。
6. **解析関数の挿入** ダイアログ ボックスには、解析関数を構成するためのパラメータがあります。 デフォルトの設定で適切な場合がほとんどですが、以下の設定を調整できます。

   パラメータ名 - 解析関数の結果を受け取るパラメータの名前を入力します。

スクリプトに出力文を挿入 - Web ページ呼び出しの後に通知のための Print 文をスクリプトに挿入するには、**Print 文** を選択します。 これによって、解析関数の結果が Silk Performer の **仮想ユーザー出力** ウィンドウに出力されます。

(Print 文によって **仮想ユーザー出力** ウィンドウに値を書き込むだけでなく) デバッグに役立つように解析結果の値を出力ファイルに書き込むには、**Writeln 文** ("write line" 文) を選択します。 出力ファイルの生成によってテストの測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全なテストでは生成しないでください。

7. **OK** をクリックします。
8. 確認のためのダイアログ ボックスが表示されます。 **OK** をクリックして、Citrix テスト スクリプトに OCR 解析関数を追加します。

生成したスクリプトの試行

TryScript では、1 人の仮想ユーザーのみが実行され、トランザクション間の思考時間または遅延が発生しないように、ストレス テスト オプションが有効になります。

注: Citrix TrueLog にはテスト (TrueLog Explorer による) の間にダウンロードしたデータの実況表示は含まれませんが、Citrix Web インターフェイス TrueLog には含まれます。 両方の TrueLog タイプには、ログ ファイル、レポート ファイルの作成、および Silk Performer Citrix Player 内での再生が含まれます。

Citrix OCR サポートに基づいて解析関数または検証関数を構成した場合、TryScript を実行する前に、OCR フォント データベースを生成する必要があります。生成しないと、これらの関数が正しく動作しないことがあります。

1. Silk Performer ワークフロー バーで、**スクリプトの試行** をクリックします。 **スクリプトの試行** ダイアログ ボックスが開きます。
2. 再生中に TrueLog Explorer 内でページ コンテンツの実況表示を参照するには (Citrix Web インターフェイス TrueLog のみ)、**TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにします。

   表示された Citrix クライアント オプション (Citrix TrueLog のみ。Citrix Web インターフェイス TrueLog では使用不可) により、TryScript の実行中に Silk Performer Citrix Player で視覚的に再生することができます。
3. **実行** をクリックします。

   注: ここでは、実際の負荷テストではなく、スクリプトのデバッグが必要かどうかを確認するためのテスト実行のみを実行します。
4. TryScript の実行が開始されます。 Silk Performer 監視ウィンドウが開き、実行の進捗について詳細情報が表示されます。

Citrix TrueLog

Silk Performer Citrix Player が Citrix TryScript の実行時に開きます。 TrueLog Explorer が Citrix Web インターフェイスの TryScript の実行時 ([TryScript] ダイアログ ボックスの **アニメーション** チェックボックスがオンになっているとき) に開きます。 TrueLog Explorer は、TryScript の実行中に実際にダウンロードされたデータを表示します。

上位の同期ノードを選択すると、最後の同期関数の後に表示されていた、再生中にキャプチャされたウィンドウのビットマップが表示されます。

ウィンドウ同期関数は、カラーの枠線で視覚化されます。 ウィンドウの作成は、緑色の枠線で示され、 ウィンドウのアクティブ化は、青色の枠線で示されます。 ウィンドウの破棄は、黄色の枠線で示されます。

TrueLog は、画面の状態を視覚化することによって、Silk Performer Citrix Player を補完します。 たとえば、Silk Performer Citrix Player のログ ウィンドウに表示されている特定のウィンドウ ID で示されるウ

ィンドウが不明な場合、それに対応する TrueLog 内の対応する同期関数を見つることによって、問題のウィンドウを示すビットマップにアクセスできます。

ユーザー入力ノード (CitrixUserInput とそれに関連する関数) には、キーボード入力とマウス入力が反映されます。 CitrixMouseClick 関数には、2 つの追跡ベクトル パラメータ (X 座標と Y 座標) があります。 赤色のひし形は、マウスをクリックした始点を示し、 赤色の十字は、マウスを解放した点を示します。 始点と終点の間の赤色の線は、マウスの経路を示します。 ボタンが押されている間に移動がなかった場合は、赤色の十字だけが表示されます。 オンスクリーン ツール チップには、補足情報 (たとえば、右クリック、左クリック、ダブルクリック) が表示されます。

値の文字列 (キーボード入力) は、対象のウィンドウ キャプションが (後続のノードで) 特定されるまで、赤色のフローティング テキストとして画面に視覚化され、文字列が入力される位置を示します。

Silk Performer Citrix Player

Silk Performer Citrix Player は、TryScript の実行が開始されたときに開き、記録されたすべてのアクションを完全なアニメーションで再生します。 マウスの移動や操作は、アニメーションのマウス アイコンでシミュレートされます。

注: Silk Performer Citrix Player は、Citrix のアプリケーションの種類でのみ開き、Citrix Web Interface のアプリケーションの種類では開きません。

Player の右上隅の **ログ ウィンドウの表示/非表示** をクリックすると、**ログ** ウィンドウが開きます。 **ログ** ウィンドウには、TryScript の実行のさまざまな側面を詳細に示す、以下の 3 つのペインがあります。

- **スクリプト** - このペインには、実行されたすべての BDL スクリプト関数および現在実行中の BDL 関数が一覧表示されます。
- **ウィンドウ** - このペインには、ウィンドウのキャプション、スタイル、サイズ、および位置など、現在のセッション中のすべてのクライアント ウィンドウのスタックが表示されます。 最上位のウィンドウは、ウィンドウ アイコンで示されており、サブウィンドウの上部に表示されます。
- **ログ** - このペインには、実行された BDL 関数やウィンドウの作成、アクティブ化、破棄を含む、すべての情報メッセージとイベントが一覧表示されます。

アクティブな関数とウィンドウは、すべてのペインで、青い矢印アイコンで示されます。

ステップ再生

1. Citrix TryScript の実行中、Silk Performer Citrix Player の右上隅にある **ログ ウィンドウの表示/非表示** ボタンをクリックすると、**ログ** ウィンドウが開きます。
2. **ステップ** ボタンをクリックします。 再生がアクティブな関数で停止します。 青色の矢印アイコンが、スクリプトの次の関数を示します。
3. **ステップ** をクリックすると、次の関数を実行します。
4. **ステップ** のクリックを続けて、スクリプト内を移動します。

   注: また、**TryScript** ダイアログ ボックスの **ステップ実行** チェックボックスをオンにして、ステップ実行を有効にすることもできます。
5. **実行** をクリックして、連続的なスクリプト処理を再開します。

タイムアウトのスキップ

Silk Performer Citrix Player は、再生中に生じたすべてのタイムアウトを待機します。 タイムアウト (デフォルトのタイムアウトは 60 秒) の待機を回避するために、**スキップ** をクリックして、待機せずに先に進みます。

注: スキップ クリックすると、スクリプトにユーザーによる中断のエラーが生成されます。

1. Citrix TryScript の実行中、Silk Performer Citrix Player の右上隅にある **ログ ウィンドウの表示/非表示** ボタンをクリックすると、**ログ** ウィンドウが開きます。

2. 再生でタイムアウトが生じたら、**スキップ** をクリックして次の関数に進めます。

ユーザー データをカスタマイズする

ユーザー データのカスタマイズでは、記録された静的ユーザー入力データを、トランザクションごとに変更されるパラメータ化された動的ユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。 このようなデータ駆動型テストの作成では、手動でのスクリプト記述は必要ありません。

Citrix ターミナル サービスによりホストされるアプリケーションに入力されるユーザー入力は、テスト中に以下の 2 つの方法でカスタマイズできます。

- **パラメータ ウィザード** によって、キーボード イベントで入力される値を指定できます。また、記録されたユーザー入力データをランダムなパラメータ化されたユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。
- 視覚的なカスタマイズによって、クリック、ドラッグ、リリースなどのマウス イベントをカスタマイズできます。

パラメータ ウィザードの使用

1. ユーザー データ入力を表す API ノードを選択します (たとえば、キーボードのデータ文字列を表す CitrixKeyString ノードを選択します)。
2. 入力データ文字列 (赤色のフローティング テキスト) を右クリックし、コンテキスト メニューから **ユーザー入力のカスタマイズ** を選択します。
3. **パラメータ ウィザード** が開きます。 **パラメータを新規作成する** を選択し、**次へ** をクリックします。
4. **パラメータ ウィザード** では、次の 2 つの方法のうち 1 つを使用して、スクリプトの値を変更できます。スクリプトの dclparam または dclrand セクションで定義された既存のパラメータを使用するか、新しいパラメータ (新しい定数値、ランダム変数、または複数列データ ファイルの値のいずれかに基づく) を作成することができます 。 新しいパラメータを作成すると、そのパラメータは既存パラメータ群に追加され、以降のカスタマイズで利用できるようになります。

   注: このタスクでは、新しいランダム変数に基づいてパラメータを作成する手順についてのみ説明します。
5. **パラメータの新規作成** ダイアログが表示されます。 **ランダム変数からパラメータを作成** ラジオ ボタンを選択し、**次へ** をクリックします。
6. **ランダム変数ウィザード** が表示されます。 テスト スクリプトに挿入するランダム変数の種類 (たとえば、ファイルからの文字列) を、ドロップ リストから選択します。 選択した変数の種類の簡単な説明が下部のウィンドウに表示されます。
7. **次へ** をクリックします。
8. **変数の名前と属性の指定** ページが表示されます。 ファイルからの文字列 というランダム変数タイプでは、指定されたファイルからランダムに選択または順次選択できるデータ文字列が生成されます。 変数の名前を **名前** フィールドに入力します。 値を呼び出す順序として ランダム または シーケンシャル を指定します。 **ファイル/名前** ドロップ リストから、構成済みのデータソース (たとえば、姓を定義する Elname) を選択します。
9. **次へ** をクリックします。
10. **使用法の選択** ページが表示されます。 新しいランダム変数を [毎回]、[トランザクションごと]、[テストごと] のどのタイミングで使用するかを指定します。
11. **完了** をクリックして、記録された値の代わりに、指定のフォーム フィールドにランダム変数を使用するように、テスト スクリプトの BDL フォーム宣言を変更します。 新しいランダム変数関数が [BDL] ビューの下部に表示されます。
12. テスト スクリプトにランダム変数関数を追加した状態で TryScript を起動し、スクリプトがエラーなしに動作することを確かめます。

マウス イベントをカスタマイズする

1. マウス アクティビティを含む CitrixMouseClick ノードを選択します。 赤色のひし形は、マウスをクリックした始点を示し、 赤色の十字は、マウスを解放した点を示します。 始点と終点の間の赤色の線は、マウスの経路を示します。 オンスクリーン ツールチップには、補足情報 (たとえば、右クリック、左クリック、ダブルクリック) が表示されます。
2. 画面上の任意の場所をクリックし、コンテキスト メニューから **ユーザー入力のカスタマイズ** を選択します。 **マウス イベントのカスタマイズ** ダイアログ ボックスが表示されます。
3. カスタマイズしたマウス移動の始点にしたい画面上の位置をクリックします。
4. カスタマイズしたマウス移動の終点にしたい画面上の位置をクリックします。
5. **カスタマイズ** ボタンをクリックしてカスタマイズを受け入れ、BDL スクリプトを変更します。

   マウス イベントのカスタマイズ結果が、記録 TrueLog のビットマップに緑色で示されます。 マウスのカスタマイズ結果は、BDL スクリプトでも緑色のテキストで示されます。 CitrixMouseClick 関数には、2 つの追跡ベクトル パラメータ (X 座標と Y 座標) があります。 このスクリプトが次回実行されたとき、ここで指定した新しい画面座標が使用されます。

テキストの同期

TrueLog Explorer には、指定したテキストやテキスト パターンが、指定した場所に表示されるまで Citrix 関数の実行を一時停止する、同期関数が用意されています。 このような同期は、検証関数を正確に実行するために重要です。

プロジェクトとシステムの設定

Citrix プロファイル設定は、Citrix の同期、ログ、仮想ユーザー シミュレーション、クライアントのオプションに関連する、プロジェクト固有の設定です。 Citrix 設定内容は、プロジェクトごとに指定されます。

このセクションでは、Citrix 再生設定に焦点をあてます。 Citrix 記録オプションは、ネットワークプロトコル、暗号化レベル、各ユーザー アクションの前に画面を記録する 設定、および RAM ディスクを使用する 設定に限られます。

Citrix XenApp オプションの構成

1. Silk Performer の **プロジェクト** ツリーで **プロファイル** ノードを展開します。
2. 構成対象のプロファイルを右クリックして、**プロファイルの編集** を選択します。

   ヒント: あるいは、メニュー バーから **設定** > **アクティブ プロファイル** を選択することもできます。

   プロファイル - [<プロファイル名>] - シミュレーション ダイアログ ボックスが開きます。左側のショートカット リストに **再生** カテゴリが表示されます。
3. ショートカット リストで下までスクロールし、**Citrix** アイコンをクリックします。 **全般** タブが開きます。
4. **同期タイムアウト** テキスト ボックスには、デフォルトの同期タイムアウトをミリ秒で入力します。これは、タイムアウト値を指定しないすべての CitrixWaitForXXX 関数で使用されます。

   デフォルト値は、60000 ms です。
5. **ウィンドウの位置を強制する** をオンにすると、記録中に取得した座標に対して移動ウィンドウ CitrixWaitForWindowCreation と CitrixWaitForWindowRestore の呼び出しを行います (デフォルトでは有効)。

このチェック ボックスがオフの場合、CitrixWaitForWindowCreation および CitrixWaitForWindowRestore の両方の関数にはパラメータ (bForcePos) があるので、呼び出すごとにこのオプションを有効にすることができます。

6. **トランザクションが終了したら切断する** をオンにすると、TInit トランザクションを含めたトランザクションが終了するごとに、Citrix クライアントを切断します (デフォルトでは無効)。
7. **セッションを正常に切断する** をオンにすると、Citrix XenApp セッションの接続を解除するときに、ログオフが実行されます (デフォルトでは有効)。
8. **各ユーザー アクションの前に画面を記録する** をオンにすると、各ユーザー アクションの開始時にもスクリーンショットをキャプチャして書き込みます (デフォルトでは有効)。

   TrueLog 生成が有効になっている場合は、同期関数それぞれの終了時にスクリーンショットを取って TrueLog に書き出します。

   CitrixWaitForScreen 関数は、画面領域をキャプチャし、記録時にキャプチャしたハッシュ値ではなく指定した条件と一致するか確認します。 このファイルを、後で、検討したり、エラー分析のために記録中にキャプチャしたものと比較することができます。

   関数呼び出し CitrixWaitForScreen が失敗し、**画面の同期に失敗したらウィンドウ領域をダンプする** がオンになっていると、キャプチャしたスクリーン領域を結果ディレクトリのファイルに書き出します。タイムアウト期間が終了するときに条件が一致しないと、関数呼び出し CitrixWaitForScreen が失敗することがあります。
9. **RAM ディスクを使用する** をオンにすると、イメージの中間ストレージとして、異なるドライブを使用します。

   注: RAM ディスクを使用する をオンにする場合は、リスト ボックスから RAM ディスクの適切なドライブ文字を選択する必要があります。 指定したドライブが RAM ディスクであれば、TrueLog 生成のパフォーマンスが向上します。
10. **シミュレーション** タブを選択します。
11. **マウス ボタンが押された状態を維持する時間** フィールドには、仮想ユーザーがマウス ボタンを押した状態でいる必要がある時間を入力します。

    デフォルト値は、200 ms です。 関数 CitrixMouseClick および CitrixMouseDblClick は、この値を使用します。
12. **ダブルクリックのクリック間の時間** フィールドには、ダブルクリックの 2 回のクリックの最大間隔を入力します。

    デフォルト値は、100 ms です。 関数 CitrixMouseDblClick は、この値を使用します。
13. **マウスの速度** フィールドには、マウスが画面を移動する速度を入力します (ピクセル/秒)。

    デフォルト値は、1000 ピクセル です。
14. **各キーが押された状態を維持する時間** フィールドには、仮想ユーザーがキーボード キーを押した状態でいる必要がある時間を入力します。

    デフォルト値は、50 ms です。 関数 CitrixKey および CitrixKeyString は、この値を使用します。
15. **文字列を入力するときのキーストローク間の時間** フィールドには、個々のキーストロークの間隔を入力します。

    デフォルト値は、100 ms です。 関数 CitrixKeyString は、この値を使用します。
16. **キー反復時間** フィールドには、繰り返し機能をシミュレートするときにキーストロークの完了に必要な時間を入力します。

    デフォルト値は、50 ms です。 50 ms の値は、1 秒間に 20 キーを表します。
17. このタブの **思考時間** 領域にある **同期に成功した後の遅延** フィールドには、仮想ユーザーが同期成功点を通過した後、非アクティブな状態でいる時間を入力します。

    デフォルト値は、1000 ms です。 関数 CitrixWaitForXXX は、この値を使用します。
18. **各ユーザー アクションの後の遅延** フィールドには、仮想ユーザーがアクション間で非アクティブな状態でいる時間を入力します。

    デフォルト値は、100 ms です。

19. **Citrix クライアント** タブをクリックして、Citrix クライアント オプションを指定します。
20. **ネットワーク プロトコル** ドロップ ボックスから、Citrix サーバーの検索と接続に使用する低レベルのネットワーク プロトコルを選択します。

    詳細については、Citrix クライアントのドキュメントを参照してください。
21. **データ圧縮を使用する** チェック ボックスをオンにすると、転送データをすべて圧縮します (デフォルトでは有効)。

    この機能によって、ファイル サイズは小さくなりますが、多くのプロセッサ リソースが必要になります。
22. **ビットマップにディスク キャッシュを使用する** チェック ボックスをオンにすると、ビットマップなどのよく使われるグラフィカルなオブジェクトを、ローカルのディスク キャッシュに保存します (デフォルトでは無効)。
23. **マウスの移動とキーストロークをキューに入れる** チェック ボックスをオンにすると、マウスとキーボードの更新をキューに入れます (デフォルトでは無効)。

    この機能によって、Citrix クライアントから Citrix XenApp サーバーに送信されるネットワーク パケット数が少なくなります。
24. **SpeedScreen 待機時間の短縮** ドロップ ボックスから、低速のネットワーク接続でのユーザー体験を向上させるために、次の項目のいずれかを選択します。

    - WAN やその他の低速の接続の場合、[オン] を選択します。
    - LAN やその他の高速の接続の場合、[オフ] を選択します。
    - 接続の待機時間に基づいて、待機時間の短縮をオンまたはオフにするには、[自動] を選択します。
25. **暗号化レベル** ドロップ ボックスから、ICA 接続の暗号化レベルを選択します。

    Citrix XenApp サーバーは、選択した暗号化レベル以上を使用できるように構成する必要があります。

    注: サーバー デフォルトまたは 基本 以外の暗号化レベルを使用すると、Citrix XenApp サーバーへの自動ログオンができなくなります。
26. 設定を保存するには、**OK** をクリックします。

全般設定を構成する

1. SilkPerformer のメイン ウィンドウのツリー ビュー領域にある [プロジェクト] タブで、[プロファイル] ノードを右クリックし、[アクティブ プロファイルの編集] を選択します。
2. [シミュレーション] タブ (再生カテゴリ) に [プロファイル - [Profile1] - シミュレーション] ダイアログが開きます。
3. 下にスクロールして Citrix アイコンを選択します。Citrix の [全般] 設定タブが開きます。
4. **全般** タブの **オプション** セクションにある **同期タイムアウト** フィールドでタイムアウト値を指定します。Citrix サーバーが遅いときは、この値を大きくすることが必要な場合があります。
5. ウィンドウの位置を強制する オプション (デフォルトでは有効) は、再生したウィンドウを CitrixWaitForWindowCreation 関数で指定された座標に自動的に移動します。
6. トランザクションが終了したら切断する オプション (デフォルトでは無効) は、Init トランザクションも含め各トランザクションの後、クライアントへの接続を解除します。
7. **全般** タブの **ログ** セクションにある 各ユーザー アクションの前に画面を記録する オプション (デフォルトでは有効) では、値が画面に実際に入力される前にユーザー入力 (データ文字列値) のオンスクリーン ディスプレイが可能になります (たとえば、ユーザーはスプレッドシートのセルに文字列の値を入力します。値は、ユーザーが **Enter** を押すまで実際には入力されません。このオプションを有効にすると、**Enter** キーのクリックのすぐ直前のノードで、文字列値が赤色のフローティング テキストとして画面に表示されます)。このオプションは、各ユーザーアクションで画面キャプチャが生成されるように指示するため、かなりの処理時間とディスク ストレージが必要です。このオプションを無効にすると、ユーザー入力の更新すべては表示されなくなります。
8. 画面の同期に失敗したらウィンドウ領域をダンプする オプションは、失敗した画面の同期すべてについて画面キャプチャが生成されるように指定します。これらのビットマップは、Silk Performer のインストールの現在の結果ディレクトリ (たとえば、RecentTryScript) にキャプチャおよび保存されている場

合、対応する記録済みの同期と比較できるためデバッグ作業に役立ちます。たとえば、1 ピクセルの違いでも、画面の同期エラーの原因に十分になりえます。このような相違は、記録された画面キャプチャと同期エラーが発生した際に取得された画面キャプチャを比較することで、視覚的に最も効果的に検出できます。

9. TrueLog の画面キャプチャの取得、読み込み、圧縮および書き込みにはかなりの処理リソースが必要となり、再生が遅くなる場合があります。RAM ディスクを使用する オプション (デフォルトでは無効) を使用すると、RAM ディスクや SSD (ソリッド ステート ドライブ) を利用できるため、従来のハード ディスクにファイルを書き込むよりも TrueLog の再生が速くなります。ドロップ ダウン リストから、読み書きの速度の速いドライブ文字を選択します。このオプションは RAM ディスクをインストールするわけではなく、選択した RAM ディスクを使用するためのものであることにご注意ください。
10. **OK** をクリックして変更を保存するか、**デフォルト** をクリックしてデフォルト設定に戻します。

シミュレーション設定を構成する

1. **プロジェクト** メニュー ツリーで **プロファイル** ノードを右クリックして、**アクティブ プロファイルの編集** を選択します。
2. **シミュレーション** タブ (再生カテゴリ) に **プロファイル - [Profile1] - シミュレーション** ダイアログが開きます。
3. 下にスクロールして Citrix アイコンを選択します。
4. **シミュレーション** タブを選択します。
5. **シミュレーション** タブの **マウス** セクションで、マウスがクリックされた状態を維持する時間、ダブルクリックのクリック間の時間、マウスの速度など、仮想ユーザーのマウス動作 (ミリ秒単位) を指定します。 シミュレートしているマウス イベントは一定の速度で移動します。 画面をジャンプして移動することはありません。
6. **シミュレーション** タブの **キーボード** セクションで、キーが押された状態を維持する時間、文字列を入力するときのキーストローク間の時間 (たとえば CitrixKeyString 関数)、キー反復時間 (CitrixKeyString 関数の irepeat パラメータ) など、仮想ユーザーのキーボード動作 (ミリ秒単位) を指定します。
7. **シミュレーション** タブの **思考時間** セクションで、同期に成功した後の遅延、および各ユーザー アクションの後の遅延について、仮想ユーザーの思考時間動作 (ミリ秒単位) を指定します。 これはユーザー反応時間の仮想シミュレーションであり、安定した再生に役立ちます。
8. **OK** をクリックして変更を保存するか、**デフォルト** をクリックしてデフォルト設定に戻します。

クライアント設定を構成する

1. **プロジェクト** メニュー ツリーで **プロファイル** ノードを右クリックして、**アクティブ プロファイルの編集** を選択します。
2. **シミュレーション** タブ ([再生] カテゴリ) に **プロファイル - [Profile1] - シミュレーション** ダイアログ ボックスが表示されます。
3. 下にスクロールして Citrix アイコンを選択します。
4. Citrix クライアント タブを選択します。
5. クライアントが動作するネットワーク プロトコル (TCP/IP または TCP/IP + HTTP) を選択します。 TCP/IP + HTTP を指定すると、ポスト コマンドを使用して HTTP プロトコルの負荷分散が行われます。 TCP/IP を指定すると、UDP が使用されます。 他のネットワーク プロトコルはサポートされません。
6. **データ圧縮を使用する** チェックボックスをオンにして、データ圧縮を有効にします (デフォルトでは有効)。
7. **ビットマップにディスク キャッシュを使用する** チェックボックスをオンにして、ハード ディスク上でのビットマップのキャッシングを有効にします (デフォルトでは無効)。

8. **マウスの移動とキーストロークをキューに入れる** チェックボックスをオンにすると、マウスの移動とキーストロークがサーバーに送信される前に、指定された時間、キューに入るようにします (デフォルトでは無効)。
9. **SpeedScreen 待機時間の短縮** を使用すると、マウス操作とキーボード操作のローカル エコーが可能になります (デフォルトでは無効)。 **ローカル エコー** とは、サーバーへのラウンド トリップを待たずに入力結果を確認できることを意味します。 オフ、オン、または 自動 を指定します。
10. クライアントの **暗号化レベル** を指定します。 オプションには、サーバー デフォルトを使用する、基本、128 ビット - ログオンのみ、40 ビット、56 ビット、および 128 ビット があります。
11. **OK** をクリックして変更を保存するか、**デフォルト** をクリックしてデフォルト設定に戻します。

エージェントで Citrix 再生を有効にする

1. Silk Performer メニュー バーの **ツール** > **System Configuration Manager** を選択します。 **System Configuration Manager** ダイアログ ボックスが表示されます。
2. *(*すべての *Citrix XenApp* クライアントのバージョンに適用*)* **アプリケーション** タブをクリックします。 Silk Performer アイコンをクリックします。 Citrix セッションを実行する権限のあるエージェント ワークステーションのユーザー アカウントに **アカウント** および **パスワード** 資格情報を入力します。
3. Citrix Recorder の **接続** ダイアログ ボックスで定義される設定を使用してアクセスされる Citrix 公開アプリケーションを記録するためには、以下を実行します。
   a) プロジェクト メニュー ツリーで **プロファイル** ノードを展開し、アクティブ プロファイルを右クリックします。 **プロファイルの編集** を選択します。
   b) 下にスクロールして **Citrix** をクリックします。
   c) **Citrix クライアント** タブをクリックします。
   d) **ネットワーク プロトコル** リスト ボックスから、TCP/IP + HTTP を選択します。

   ICA ファイルを利用する Citrix 公開アプリケーションを記録するには、ICA ファイルの [WFClient] セクションで、TcpBrowserAddress= を HttpBrowserAddress= に変更します。

OCR のシステム設定

TrueLog Explorer で、解析関数や検証関数用の OCR を有効にするには、 Silk Performer のシステム設定を使用して、フォント データベースを生成する必要があります。

OCR は、ビットマップ内のフォントやテキスト スタイルを認識するために、フォント (またはパターン) データベースを使用します。 デフォルトのフォント セットでほとんどのシナリオはカバーできますが、状況によっては、ほかのフォントやフォント スタイルの追加が必要になることがあります。 新しいフォントを追加したり、システムからフォントを削除したときは、毎回、新しいデータベースを生成する必要があります。

データベース内のフォントが多すぎると、処理が遅くなったり、読み取り間違いをする場合があります。このため、テキスト文字列の取得対象となるビットマップで使用されるフォントのみを含めることをお勧めします。

OCR 用のシステム設定を構成する

1. Silk Performer 内で 設定 > システム... に移動して Citrix アイコンを選択します。
2. **OCR** タブで、**追加 >>** または **すべて追加** ボタンを使用して、OCR で使用するフォントを **システム フォント** リスト ボックスから **選択済みのフォント** リスト ボックスに移動します。
3. **選択済みのフォント** リスト ボックスから不要なフォントを削除するには、**すべて削除** または **<< 削除** ボタンを使用します。
4. **サイズ** フィールドで、使用するフォント サイズの範囲を指定します (たとえば、8-20)。

5. イタリック、太字、および 下線 の各チェック ボックスをオンにして、含めるフォント スタイルを定義します。メニュー エントリを検証する場合は、下線 チェック ボックスがオンになっていることを確認します。
6. **フォント データベースの生成** をクリックします。
7. **フォント ベースの構築** ダイアログ ボックスが開きます。**OK** をクリックして、既存のフォント データベースを新しいデータベースで置き換えることを確認します。

   注: フォント データベースの作成中に問題が発生した場合は、データ実行防止の問題である可能性があります。この問題を解決するには、以下のステップを実行します。
8. Silk Performer の [システム設定] ダイアログ ボックスで、**OK** をクリックして、変更を受け入れます。

テストのベスト プラクティス

GUI ベースのテストは多くの課題がありますが、現実のユーザー体験を綿密にシミュレートする機会となるため有益です。 Silk Performer による GUI ベースのテストでは、ターミナル サービス プロトコルをシミュレートしてアプリケーションに負荷をかけます。

Citrix テストを実行するには、少なくとも 2 つの物理コンピュータ (ターミナル サービス環境とテスト対象のアプリケーション (AUT) を実行するコンピュータ、および Silk Performer と Citrix クライアント ソフトウェアを実行するコンピュータ) を使用する必要があります。

負荷生成コンピュータは、ターミナル サービス ネットワーク プロトコルを大量に作成して、多くのユーザーをシミュレートします。 ターミナル サービス サーバー (テスト対象のシステム) は、MS Windows ベースの複数のデスクトップ セッションを同時にシミュレートします。 このサーバーは、高い負荷のかかっている GUI アプリケーションもホストします。

負荷生成コンピュータとテスト対象のシステムの間に、切り離したネットワーク (LAN) を配置することをお勧めします。 これにより、ネットワークのスループット分析が可能になります。 また、切り離したネットワークは外的影響を受けにくいため、エラーおよびネットワークの紛らわしいスループット結果が減少します。

テスト準備

- ユーザー インターフェイス (UI) 設計 - アプリケーションの UI 設計に影響を及ぼすことのできる機会があれば、うまく利用しましょう。UI を一貫性のある確かなものにすれば、再利用可能なサブルーチンを作成しやすくなります。それによりテスト コードの作成を減らすことができ、その結果、維持する必要のあるコードが少なくなります。
- テスト計画 - 十分に検討されたテスト計画は、GUI ベースの 負荷 テストの成功の前提条件です。テスト計画では、目標、目的、テストの実行、重要な指標などを定義する必要があります。
- ユース ケース - ユース ケースは、テストを計画する、ステップ by ステップのシナリオです。ユース ケースは、たとえば、アプリケーションの起動、プロジェクトを開く、設定の編集など、典型的なユーザー-アプリケーション間の対話を表します。ユース ケースに従って、記録プロセスを進めます。
- アプリケーションのエキスパート - 十分に文書化されたユース ケースがあっても、アプリケーションのワークフローのすべてを理解できないこともあります。テスト対象のアプリケーションに関して、問い合わせられる専門知識を持つ担当者 (たとえばビジネス プロセス オーナー) がいると助けとなります。
- 画面解像度 - 画面解像度が高ければ高いほど、各ターミナル セッションに対するシステム要件は厳しくなります。記録と再生には、画面解像度を 800x600 以下にすることをお勧めします。このアプローチは、古いコンピュータとの互換性があり、負荷 テストに含めることもできます。
- リモート ユーザー - リモート ユーザーがアプリケーションを利用できるようにする場合、一部の 負荷 テスト中に低速のネットワークのシミュレーションを行うことを検討します。低速のネットワーク接続はアプリケーションのパフォーマンスに大きく影響するため、ネットワークが混雑した状態をシミュレーションして行う追加テストは、テスト結果の向上に役立ちます。これを行うには、Silk Performer のネットワーク エミュレーション機能を使用します。

目標の定義

負荷 テストの計画で最初にすべきことは、目標を定義することです。テスト担当者はよく、目標はすべての関係者にとって明確であると考えますが、多くの場合そうではありません。品質保証チームと管理者が同じ目標に取り組むことが重要です。

目標を記述する際は、まず、次のような最も高いレベルの目標を検討します。

1. アプリケーションのベースライン パフォーマンスを特定する
2. アプリケーションを最適化/調整する
3. テスト対象のアプリケーションが本稼働の準備ができているかどうかを判断する

高レベルの目標を設定した後、それらの目標を、どのように目標を達成および測定するか、どのような結果が許容されるかなどの問題をカバーする、明確で測定可能な目的に細分化する必要があります。

以下では、テスト目標をどのように細分化するかについて例示します。

目標 1：アプリケーションのベースライン パフォーマンスを特定して文書化する

- Silk Performer を使用して、重要なアプリケーション関数のレスポンス時間を測定します。
- ウィンドウ/画面の同期にタイマ関数を設定します (WaitFor イベント)。これらの測定値は、最終レポートに表示されます。

重要なタイマ：

- **ログイン** ウィンドウで **OK** ボタンをクリックした時点から、**ようこそ** ウィンドウが表示されるまでにかかった時間を測定します。
- **クエリ結果** ボタンをクリックした時点から、入力したリストが表示されるまでにかかった時間を測定します。

目標 2：アプリケーションを最適化/調整する

- アプリケーションの最適化/調整を 5 日間行います。
- すべての変更内容と、テスト対象のアプリケーションのパフォーマンスへのそれらの影響を文書化します。

目標 3：テスト対象のアプリケーションが本稼働の準備ができているかどうかを判断する

- すべてのレスポンス時間が 5 秒を下回るという条件が満たされたら、アプリケーションは本稼働が可能です (Silk Performer では、各ウィンドウと画面の同期に関して指標を提供しています) 。

測定可能な目的のリストに従い、テスト結果は、アプリケーションが本稼働できる明確な時点を定義します。定義した目標へのアプリケーションの調整が適切であれば、これにより、アプリケーションを際限なく最適化するというリスクも排除されます。

ユース ケースを作成する

ユース ケースとは、テスト対象のアプリケーションで作業する場合にユーザーが行う標準的なタスクです。ユース ケースは、テストを必要とするアプリケーションの機能を使用する必要があります。 適切に動作している重要な機能のみをテストすることが不可欠です。 この時点で、機能テストはすでに完了しているべきであり、実施しません。 テストのプロセスは長く、ユース ケースが長ければ長いほど、テストにかかる時間も長くなります。

ユース ケースを段階的に実行する場合には、重要な画面イベントをすべて書き出します。 たとえば、Microsoft Excel で式を入力する場合、式の処理のためにセルの変更を文書化する必要があります。 このようなイベントは、スクリプトの開発中に重要となる画面の同期に置き換えられます。 テキスト同期は、テキスト ベースの画面の同期に使用できます。

ユース ケースを詳細レベルまで文書化します。 すべてのマウス クリック、キー ストローク、および予想結果を文書化する必要があります。 これは最初は面倒かもしれませんが、テストケースの記録の開始時に作業が容易になります。

たとえば、単に Microsoft Word の既存のインスタンスの場所を特定したり、ドキュメントを開いたりするテストでは、ユース ケースにあるべき詳細レベルが表示されます。 角括弧 ("[ ]") はイベントを示します。

```
In the “Microsoft Word” window, navigate to the File menu and select Open….
[The “Open” dialog box opens]
Select ‘Test.doc.’
Click Open.
[The “Open” dialog box closes]
[The “Microsoft Word” window has focus once again]
```

上記の例では、ウィンドウのタイトルがそのまま文書化されます。 このような情報は、スクリプトをアプリケーションと同期した状態に維持するうえで必要なため、以降のスクリプトの開発時に重要となります。ユース ケースが十分に文書化されていると、アプリケーションのスクリプト化が容易です。

ユース ケースの記述には、いくつかの方法があります。 使い慣れた XML 表記、番号の付けられた表記、または他の形式を使用できます。

XML ユース ケースの例

```
<Task Name=”Open a document”>
  In the “Microsoft Word” window, navigate to the File menu and select Open….
  [The “Open” dialog box displays]
  Select ‘Test.doc.’
  Click Open.
  [The “Open” dialog box closes]
  [The “Microsoft Word” window has focus once again]
</Task>
```

番号の付けられたユース ケースの例

```
100.01 – Open a document
  In the “Microsoft Word” window, navigate to the File menu and select Open….
  [The “Open” dialog box displays]
  Select ‘Test.doc.’
  Click Open.
  [The “Open” dialog box closes]
  [The “Microsoft Word” window has focus once again]
100.02
    …
```

安定した環境を確保する

記録を開始する前に、テスト対象のアプリケーションの UI が、少なくともテストが完了するまで追加変更されないことを確認してください。 このためには、開発チームと十分に連絡を取り合う必要があります。テスト スクリプトの記録後に UI が変更されると、スクリプトが使用できなくなる可能性があります。

また、テストで使用するデータベースをバック アップして復元できるようにしてください。 データベース コンテンツへの変更は多くの場合、UI コンテンツにおける違い (たとえば、リストの項目の数が多いまたは異なる、クエリ結果が異なるなど) につながるため、各テスト サイクルの開始前にデータベースをベースライン状態に戻す必要があります。

トラブルシューティング

ユース ケースを文書化すると、テスト スクリプトの記録またはコーディングは相対的に単純なタスクとなります。 Silk Performer には、記録/再生オプションとテスト スクリプトの手動コーディング オプションの両方が備えられています。 各アプローチには、長所と短所があります。 Silk Performer でテスト スクリプトを作成する最も効率的な方法は、次のように、2 つのアプローチを組み合わせることです。

1. Silk Performer を使用して、ユース ケースの説明で示したシナリオを記録します。
2. TrueLog Explorer を使用して、スクリプトを視覚的にカスタマイズします (ここで、“テキスト同期” 機能が 動作します)。
3. より複雑なスクリプト作成については、生成された BDL スクリプトを手動で編集します。

   手動でのスクリプト編集は、ユース ケースのコンテンツをテスト スクリプトに挿入 (コメントの形式で) する際の最良のオプションでもあります。 以下の例を参照してください。この例では、番号の付けられた表記形式を使用しています。

```
100.001 In the “Microsoft Word” window, navigate the menu to File, Open….
100.002 [The “Open” dialog window shows]
100.003 Select Test.doc.
100.004 Click Open.
100.005 [The “Open” dialog window goes away]
100.006[The “Microsoft Word” window has focus again]
```

4. 次のように、これらのコンテンツを BDL スクリプトにコピー/貼り付けできるようになりました。

```
// 100.001 In “Microsoft Word" window,
     hwndWordMainWindow := CitrixSearchWindow(“*Microsoft
     Word”, MATCH_Wildcard);
     CitrixWindowBringToTop(hwndWordMainWindow);

// navigate the menu to File, Open….
     CitrixKey(KEY_Alt);
     CitrixKeyString("f");
     CitrixKeyString("o");

// 100.002 [The “Open” dialog window shows]
     hwndOpenDialog := CitrixWaitForWindowCreation(“Open”,
     Match_Exact);

// 100.003 Select Test.doc.
     CitrixMouseClick(150, 100, hwndOpenDialog, MOUSE_
     ButtonLeft);

// 100.004 Click Open
     CitrixMouseClick(300, 200, hwndOpenDialog, MOUSE_
     ButtonLeft);

// 100.005 [The “Open” dialog window goes away]
     CitrixWaitForWindow(hwndOpenDialog, EVENT_Destroy);

// 100.006 [The “Microsoft Word” window has focus again]
     CitrixWaitForWindow(hwndWordMainWindow, EVENT_Activate);
```

   スクリプトが失敗した場合、ユース ケースに関してどこでスクリプトが失敗したのかを判別する際に、これらのコメントが役立ちます。

記録のヒント

ターミナル サービス スクリプトを記録する際は、再生中の対象のウィンドウの位置とサイズが予定外のイベントにより変更され、テストの実行が失敗する可能性があることを認識する必要があります。 ここでは、そのような再生エラーを探して修正するという作業を行わなくてすむようにする、全般的なヒントを提供します。

Windows の **ファイル名を指定して実行** コマンドで、アプリケーションを起動します。 デスクトップ アイコンから、またはマウスをクリックしても、アプリケーションを起動できますが、アプリケーションを異なるサーバーに移動する場合、または GUI がユーザー間で異なる場合は、デスクトップ アイコンは別の場所に表示される可能性があることを考慮してください。 このため、Windows の **ファイル名を指定して実行** コマンド ウィンドウで、アプリケーションの EXE ファイルへのパスを入力することをお勧めします。

キーボードのショートカット

マウス クリックでは、x/y 座標、ウィンドウのリファレンスが必要となり、対象のウィンドウには、フォーカスがあてられている必要があります。 マウス クリック追跡のための GUI テストのテクノロジは信頼性が高まりましたが、その一方で、Windows は動的に変化する環境のままです。このため、小さな不整合を許容できるスクリプトを構築する必要があります。 このための方法の 1 つは、キーボードのショートカットを使用することです。 ショートカットでの唯一の要件はウィンドウのフォーカスです。

注: Citrix Recorder からフォーカスがはずれてしまうため、**Windows ロゴ** キーはサポートされていません。

Windows アプリケーションのほとんどすべてのメニュー システムで、**Alt** キーの組み合わせを使用して特定のコマンドを開始することができます。 **Alt** キーのショートカットの使用を記録するには、まずアプリケーションにフォーカスをあて、次に **Alt** キーを押します。これにより、メニュー システムがフォーカスされます。 メニュー項目の多くには、下線付きの文字が 1 つ付与されています。 これは、**Alt** キー コマンドが特定のメニュー項目に利用可能であることを示しています。 適切な文字キーを押してサブメニューにアクセスし、開始するメニュー項目まで移動します。 メニュー項目自体にキーの組み合わせがある場合もあります。 たとえば、Microsoft Word で文書を開くには、Ctrl+O の組み合わせを使用します。 このショートカットでは、メニューがフォーカスされている必要はありません。

ウィンドウを最大化する

キーボードのショートカットが利用できず、マウスを使わなくてはならない場合は、作業中のウィンドウを最大化することで、不正なマウス クリックをする可能性を軽減できます。 これにより、毎回ウィンドウが決まった位置に固定され、マウス クリックはより正確になります。 原則として、ウィンドウの最大化が可能な場合は最大化することをお勧めします。

注: Alt-Space、X が、アクティブなウィンドウを最大化する WIndows のキーボードの組み合わせです。

セッションの記録中、ウィンドウを最大化できない場合は、クリックするまえにウィンドウをデスクトップの左上隅に移動します。 記録時に、Recorder がフォアグラウンドにあるウィンドウのハンドルをつかめないことがあります。そのため、座標をデスクトップに対する相対座標にすることが必要です。

システムをクリーンに維持する

ターミナル サービス セッション中に予期しないイベントが起こらないようにするために、Windows のデスクトップをできる限り整理した状態を維持してください。 たとえば、スクリプトの再生中に Windows ネットワークのメッセージが表示されるとします。 その後のマウス クリックが、意図したアプリケーション ウィンドウではなく、Windows ネットワークのメッセージ ウィンドウに対して行われると、スクリプトが動作しなくなります。

適切なフォーカスを設定する

ウィンドウをクリックする前に、クリックするウィンドウが存在し、フォーカスが設定されているか確認します。 以下は、このプロセスの自動化を支援するサンプルの BDL 関数です。

```
function  MyCitrixWaitForWindowCreationAndActivation(sCaption:string;
                    nMatch      :  number  optional;
                    nStyle      :  number  optional;
                    nX          :  number  optional;
                    nY          :  number  optional;
                    nWidth      :  number  optional;
                    nHeight     :  number  optional
                    )  :  number
var
     hwndNewWindow   :  number;   //  hwnd  of  the  new  window  created
     nRTT:number;
begin
     //
     //  Check  if  window  exists
```

```
    //
        hwndNewWindow := CitrixSearchWindow(sCaption, nMatch);
          hwndNewWindow := 0;
    //
    // If window doesn't exist, wait for creation
    //
        if (hwndNewWindow < 1) then
            hwndNewWindow := CitrixWaitForWindowCreation(sCaption, nMatch,
            nStyle, nX, nY, nWidth, nHeight);
        end;
    //
    // Check if window is already active, wait if it's not active
    //
        MyCitrixWaitForWindowActivate(hwndNewWindow);
        MyCitrixWaitForWindowCreationAndActivation:=hwndNewWindow;

end MyCitrixWaitForWindowCreationAndActivation;
```

パラメータを使用する

2 度以上同じ情報を入力すると、テスト対象のシステムは、同じ作業を繰り返すのではなく、 データをメモリにキャッシュします。 このため、作業ができるだけ現実的なものとなるように、入力/リクエスト データを変更することが重要です。 たとえば、異なるデータ範囲を有する異なるユーザー アカウントを使用します。 これは、テストに CSV (コンマ区切り値) ファイルを入力として使用することで実現できます。

検証を追加する

検証とは、サーバーから受信されたレスポンスが正しいかどうかを確認するためにコードに追加されるチェックです。 GUI ベースのテストを実施する場合、検証はウィンドウの同期により自動的にスクリプトに追加されますが、コードに検証をさらに追加することをお勧めします。

タイマを追加する

カスタム タイマは、アプリケーションのレスポンス時間の追跡のために非常に重要です。 カスタム タイマがないと、アプリケーションに対する総合的なエンド ユーザー体験を判断することができません。 指標は、アプリケーションのどの点がよく実行されており、どの点がそうでないのかを判断する際に役立ちます。

テストスクリプトにカスタム タイマを追加する前に、アプリケーションの重要な領域を特定して、MeasureStart および MeasureStop 呼び出しを行う場所を決定します。 この作業については、TrueLog Explorer のログが参考になります。

以下は、Silk Performer のスクリプトでタイマを使用している例です。

```
//
// Submit a work order.
//
CitrixMouseClick(27, 31, hwndWorkOrder, MOUSE_ButtonLeft);

//
// Start the response time clock.
//
MeasureStart("202.01: Work Order Submission.");

//
// Wait for the "Order Submission Complete" dialog box.
//
MyCitrixWaitForWindowCreationAndActivation(
     "Order Submission Complete",
     MATCH_Exact
     );
```

```
//
// Stop the response time clock.
//
MeasureStop("202.01: Work Order Submission ");
```

スクリプトのデバッグ

Silk Performer Citrix での再生中に、記録中にキャプチャされた値と異なる値が検出されると、ウィンドウがアクティブにならなかったり、画面の同期に失敗することがあります。 同期に関する問題の原因は、常に明白とは限りません。 画面の位置が 1 ピクセルだけ変化したことが原因の場合もあります。

画面の同期の失敗よりもよくあるのは、再生中にウィンドウがアクティブにならないことです。 このような場合は、それに対応するユーザー アクションに関連するスクリーンショットが失敗の原因を究明する手がかりになることがあります。

注: ユーザー エラーがないのに、ウィンドウがたまにしかアクティブにならないこともあります。 このような場合は、関連する CitrixWaitForWindow 関数を削除する必要があります。

Silk Performer Citrix の再生では、エラーが発生すると画面をキャプチャし (デフォルトの設定)、そのビットマップをディスクに書き込みます。 デフォルトでは、スクリーンショットはレコーダによってプロジェクト ディレクトリ内のスクリーンショット ディレクトリに書き込まれます。 再生では､現在使用されている結果ディレクトリにスクリーンショットが保存されます。 ビットマップ表示プログラムを使用して、記録画面と再生画面を視覚的に比較できます。

ビットマップをキャプチャして保存するには、Silk Performer の **画面の同期に失敗したらウィンドウ領域をダンプする** Citrix オプションをアクティブにする必要があります (デフォルト)。

トラブルシューティング スクリプト

負荷テストの実行時に、タイムアウト エラーやその他の実行失敗が発生することがあります。 ここでは、そのようなエラーを回避、検索、修正するヒントを提供します。

TrueLog Explorer を使用する

テスト実行が予期しないダイアログ ボックスが表示されることで失敗し、それが原因でスクリプトが作業中のウィンドウへのフォーカスを失うことがしばしばあります。 そのため、テストを実行する前に Silk Performer の TrueLog On Error 機能を有効にしておくことをお勧めします。 そうすると、エラーが発生した場合に、TrueLog Explorer で視覚的に割り出すことができます。

ターミナル サーバー セッションをクリアする

テスト実行が失敗し後は、テストを再開する前にすべてのターミナル サーバー セッションをリセットしてください。リセットしない場合はスクリプトが失敗します。

アプリケーション エラーを処理する

テスト中、生成された負荷が原因で、アプリケーションによってエラーが呼び出されることがあります。エラーの処理および報告を行うイベント ハンドラーをスクリプトに追加しておくと便利です。

ここでは、**Program Error Intercepted** という名前のウィンドウを継続的に監視し、そのようなウィンドウが表示された場合に ALT-C を実行してウィンドウを閉じるイベント ハンドラーの例を示します。 またそのようなエラーが発生すると、イベント ハンドラーはエラー メッセージを生成します。

```
dclevent
  handler Handler1 <EVENT_CITRIXINTERRUPT>
  var
    nInterrupt, nWindow : number;
    nStyle              : number;
    sWindowCaption      : string;
```

```
    begin
       CitrixGetActInterrupt(nInterrupt, nWindow);
       ErrorAdd(FACILITY_CITRIXENGINE, 47, SEVERITY_INFORMATIONAL);
       print(string(nWindow));
       CitrixGetWindowCaption(nWindow, sWindowCaption);
       if sWindowCaption = "Program Error Intercepted" then
          CitrixKey(67, MOD_Alt); // 'c'
       end;
       ErrorRemove(FACILITY_CITRIXENGINE, 47);
    end Handler1;
```

思考時間を回避する

テスト対象のアプリケーションのスクリプトがオーバーランするのを回避するために Wait() 文または ThinkTime 文を使用しているかもしれませんが、以下の 2 つの理由からこの方法はお勧めできません。

- 負荷の生成が増加すると、アプリケーションの処理速度が著しく遅くなる場合がある。 結果的に待機文が短すぎ、アプリケーションのオーバーランの問題自体が再度発生することがある。
- ウィンドウ、テキスト、画面の同期のレスポンス時間を計測する場合は、Wait() 文の時間によってレスポンス時間が不自然に長くなってしまうことがあります。

これを解決するには、同期を使用します。

スクリプトの一部を記録し直す

アプリケーションの動作を変更すると、記録したスクリプトの一部が使用できなくなることがあります。ユース ケースの一部を記録し直すというのも 1 つの手段ですが、レコーダが既存のウィンドウのハンドルを追跡できず、結果としてウィンドウの処理番号が間違ってしまうということあるため注意が必要です。こういった問題のせいで、新しく記録したスクリプトと元のスクリプトの統合が極めて煩雑になってしまうことも考えられます。 ユース ケースが小さい場合は、ユース ケース全体を記録し直すことをお勧めします。 記録し直さない場合は、以下に概説されている処理に従って、新しいスクリプトと古くなったスクリプトを統合してください。

1. 記録し直すコードのセクションをコメントアウトします。
2. 置換が必要なコード セクションの場所と、ユース ケース内の対応するセクションを一致させます。
3. ターミナル サービス セッションを、ユース ケース内の対応するポイントまで移動させます。
4. オープン ターミナル サービス セッションの記録を開始します。
5. 置換が必要なユース ケースのアクションを実行します。
6. レコーダを停止します (その後記録のスクリプトが作成されます)。
7. 新たに記録されたスクリプトのウィンドウ ハンドル変数を、それぞれ元のスクリプトのハンドルと置き換えます。 CitrixSearchWindow() などの関数を使用すると、正しいウィンドウ ハンドルを特定できます。
8. 編集したコードを元のスクリプトにコピーします。

Citrix ダイアログ ボックスを処理する

Citrix ターミナル サービス セッションへの接続方法およびライセンス セットアップにより、次のダイアログボックスの両方またはいずれかが表示されます。

- ICA シームレス ホスト エージェント
- Citrix ライセンスの警告

これらのダイアログ ボックスは情報であり、最初にターミナル サービス セッションにログインしたときに表示される場合と表示されない場合があります。これらのダイアログ ボックスの処理方法には、次の 2 つがあります。

Citrix ダイアログ ボックスを処理する (解決策 1)

この解決策では、ダイアログ ボックスが表示された場合にそのダイアログ ボックスを処理する割り込みを作成します。

```
transaction TMain
  var
  begin
    CitrixInit(800, 600);
    CitrixAddInterrupt(INTERRUPT_WindowCreate, "ICA Seamless Host Agent",
MATCH_Exact);
    CitrixConnect("lab74", "labadmin", "labpass", "testlab1", COLOR_16bit);
    CitrixWaitForLogon();
    hWnd4 := CitrixWaitForWindowCreation("", MATCH_Exact, 0x96840000, -2, 572, 804,
30);
    CitrixWaitForWindowCreation("Program Manager");
    CitrixMouseClick(36, 17, hWnd4, MOUSE_ButtonLeft, MOD_None, -1, 0);
    hWnd11 := CitrixWaitForWindowCreation("", MATCH_Exact, 0x96400000, 2, 313, 163,
263);
    CitrixMouseClick(62, 247, hWnd11, MOUSE_ButtonLeft);
    CitrixWaitForWindow(hWnd11, EVENT_Destroy);
    hWnd12 := CitrixWaitForWindowCreation("Shut Down Windows", MATCH_Exact,
0x94C808CC, 191, 136, 417, 192);
    CitrixWaitForWindow(hWnd12, EVENT_Activate);
    CitrixMouseClick(203, 170, hWnd12, MOUSE_ButtonLeft);
    CitrixWaitForDisconnect();
  end TMain;

dclevent
  handler Handler1 <EVENT_CITRIXINTERRUPT>
  var
    nInterrupt, nWindow : number;
    nStyle              : number;
  begin
    CitrixGetActInterrupt(nInterrupt, nWindow);

    ErrorAdd(FACILITY_CITRIXENGINE, 47, SEVERITY_INFORMATIONAL);
    print(string(nWindow));
    if CitrixGetWindowStyle(nWindow, nStyle) and (nStyle <> 0xB4000000) then
      CitrixWaitForWindow(nWindow, EVENT_Activate);
      CitrixMouseClick(201, 202, nWindow, MOUSE_ButtonLeft);
      CitrixWaitForWindow(nWindow, EVENT_Destroy);
    end;
    ErrorRemove(FACILITY_CITRIXENGINE, 47);

  end Handler1;
```

Citrix ダイアログ ボックスを処理する (解決策 2)

このサンプル コードは、Citrix ダイアログ ボックスが表示されるまで 30 秒間ループします。ダイアログ ボックスが表示されると、このコードによりダイアログ ボックスが閉じられます。

```
function MyCitrixStartup(nMaxWait: number optional): boolean
    var
        hwndICAHandle          : number;
        hwndFoundLicenseWarning : number;
        nCount                 : number;
    begin
        hwndICAHandle:=-1;
        hwndFoundLicenseWarning:=-1;
        nCount:=0;
```

```
        if (nMaxWait = 0) then
            nMaxWait:=10;
        // if no wait time was passed,
        // use 10 tries (seconds) as a default
        end;

        MeasureStart("MyCitrixStartup");

    //
    // loop until we've handled the conditions or we've tried
    //

        while ((nCount < nMaxWait) and ((hwndICAHandle <=0) or
            (hwndFoundLicenseWarning <=0))) do

    //
    // Just a little feedback, every 10 tries
    //
            if ((nCount MOD 10) =0) then
                print(string(nCount) + ")MyCitrixStartup "
                + " vUser:" + string(GetUserId())
                + " hwndICAHandle=" + string(hwndICAHandle)
                + " hwndFoundLicenseWarning="
                + string(hwndFoundLicenseWarning),
                OPT_DISPLAY_ERRORS , TEXT_GREEN );
            end;
    //
    // if we haven't handled this window yet
    //
            if (hwndICAHandle  <=0)
then
                hwndICAHandle := CitrixWaitForWindowCreation
                ("ICA Seamless Host Agent", MATCH_Exact, 0x94C800C4,
                0, 0, 0, 0, false, 1, true);

                if (hwndICAHandle > 0) then
                    if (CitrixWindowBringToTop(hwndICAHandle)) then
                        CitrixKey(KEY_ENTER); // press ok to close the dialog
                        CitrixWaitForWindow(hwndICAHandle, EVENT_Destroy);
                        // wait for the close
                    end; // end waiting for window to top
                end; // end if we have a valid handle
            end; // if window has not been found yet

            if (hwndFoundLicenseWarning  <=0)
then
                hwndFoundLicenseWarning := CitrixWaitForWindowCreation
                ("Citrix License Warning Notice", MATCH_Exact, 0x94C800C4,
                0, 0, 0, 0, false, 1, true);

                if (hwndFoundLicenseWarning > 0) then
                    if (CitrixWindowBringToTop(hwndFoundLicenseWarning)) then
                        CitrixKey(KEY_ENTER);  // Press ok
                        CitrixWaitForWindow(hwndFoundLicenseWarning,
                            EVENT_Destroy);
                        // wait for it to go away
                    end; // end waiting for window to top
                end; // end if we have a valid handle
```

```
            end; // if window has not been found yet

            nCount :=nCount+1;
            Wait 1.0;
        end; // while nCount

        MeasureStop("MyCitrixStartup");
        //
        // return true if we handled any one of these conditions
        //

        MyCitrixStartup2 := (hwndFoundLicenseWarning > 0)
            or (hwndICAHandle > 0) ;

          print("MyCitrixStartup "
            + " vUser:" + string(GetUserId())
            + " Waited " + string(nCount) + " of " + string(nMaxWait)
            + " hwndICAHandle=" + string(hwndICAHandle)
            + " hwndFoundLicenseWarning=" + string(hwndFoundLicenseWarning),
            OPT_DISPLAY_ERRORS , TEXT_GREEN );

end MyCitrixStartup;
```

索引

B

C

O

T

W

え

か

き

く

さ

し

す